第四个与其他的不同，是自己动手分析，类似于kaggle

```
train.shape
```

```
(8000, 140)
```

```
y=train["happiness"]
y.value_counts()
```

```
 4    4818
 5    1410
 3    1159
 2     497
 1     104
-8      12
Name: happiness, dtype: int64
```

首先分析了数据集，shape是8000*140，其中139个为特征，“happiness"是分类标签，查看了一下分布，发现有一个-8的分类，看了一下index文档后发现是拒绝回答，由于样本所占比例很小，所以我决定直接删除

```
In [4]: num=[]
        for i in range(len(y)):
            if y[i]==-8:
                num.append(i)
```

```
In [5]: num
Out[5]: [609, 1064, 1419, 1702, 2700, 2884, 3058, 3198, 3946, 5619, 5896, 7081]
```

```
In [6]: train_del=train.drop(train[train['happiness']==-8].index)
```

```
In [7]: train_del.shape
Out[7]: (7988, 140)
```

```
In [10]: y_del=y.drop(num)
```

```
In [11]: y_del.shape
Out[11]: (7988,)
```

删除后共有7988行

```
In [20]: train_del["survey_age"]=2015-train_del["birth"]
         test["survey_age"]=2015-test["birth"]
         train_del.head()
Out[20]:
```

rvice_1	public_service_2	public_service_3	public_service_4	public_service_5	public_service_6	public_service_7	public_service_8	public_service_9	survey_age
50	60	50	50	30.0	30	50	50	50	56
90	70	70	80	85.0	70	90	60	60	23
90	80	75	79	80.0	90	90	90	75	48
100	90	70	80	80.0	90	90	80	80	72
50	50	50	50	50.0	50	50	50	50	21

```
In [ ]:
```

增加了新的一列特征，被调查时的年龄，通过分析数据集发现调查是2015进行的，一次年龄就用2015-出生年

```
In [24]: data_null = train_del.isnull().sum()/len(train_del) * 100
         data_null

survey_type        0.000000
province           0.000000
city               0.000000
county             0.000000
survey_time        0.000000
gender             0.000000
birth              0.000000
nationality        0.000000
religion           0.000000
religion_freq      0.000000
edu                0.000000
edu_other         99.962444
edu_status        14.021032
edu_yr            24.661993
income             0.000000
political          0.000000
join_party        89.697046
floor_area         0.000000
property_0         0.000000
property_1         0.000000
```

看了一下有一些列包含大量的空值，删除控制数量大于80%的列

```
In [13]: null_list=['edu_other','join_party','property_other','invest_other']
         train=train_del.drop(null_list, axis=1)
         test=test.drop(null_list, axis=1)
```

```
In [15]: # 建立一个每一个特征值可能取值所占比例最大的列表，并显示他们所占的百分比
         sk_df = pd.DataFrame([{'column': c, 'uniq': train[c].nunique(), 'skewness': train[c].value_counts(normalize=True).values[0] * 100} for c
         sk_df = sk_df.sort_values('skewness', ascending=False)

         # 将结果输出
         print(sk_df)

          column  uniq    skewness
83      invest_6     1  100.000000
85      invest_8     2   99.924887
84      invest_7     2   99.924887
82      invest_5     2   99.812218
24    property_6     2   99.599399
..           ...   ...         ...
136   survey_age    77    2.478718
8          birth    77    2.478718
5         county   130    1.014021
6    survey_time  7218    0.050075
0             id  7988    0.012519

[137 rows x 3 columns]
```

```
In [18]: drop_feature=[]
         for i in range(len(sk_df)):

             if sk_df['skewness'][i]>99:
                 drop_feature.append(sk_df['column'][i])
         train.drop(drop_feature, axis=1, inplace=True)
         test.drop(drop_feature, axis=1, inplace=True)
         train

Out[18]:
    id  happiness  survey_type  province  city  county  survey_time  gender  birth  nationality  ...  public_service_1  public_service_2  public_service_3  pu
```

skeness>99对于判断分类基本上没有用，去掉

```python
In  [18]: def income_cut(x):
              if x<0:
                  return 0
              elif  0<=x<1200:
                  return 1
              elif  1200<x<=10000:
                  return 2
              elif  10000<x<24000:
                  return 3
              elif  24000<x<40000:
                  return 4
              elif  40000<=x:
                  return 5



          train["income_cut"]=train["income"].map(income_cut)
          test["income_cut"]=test["income"].map(income_cut)
          train.drop(["income"], axis=1,inplace=True)
          test.drop(["income"], axis=1,inplace=True)
```

```python
In  [19]: train.drop(["id"], axis=1,inplace=True)
          train.drop(["happiness"], axis=1,inplace=True)
          test.drop(["id"], axis=1,inplace=True)
```

对于收入进行分类

之后就是使用模型进行预测了，不赘述了

最后提交结果score0.467，排名不高，由于时间问题只是简单预测，日后可以更认真地分析来提升效果。